Κατανεμημένα Συστήματα 1

$2^{η}$ Εργασία

Δούρου Βασιλική Ευαγγελία

A.M.:1072633

Ερώτημα 1:

Ο ζητούμενος ασύγχρονος κατανεμημένος αλγόριθμος που θα εκλέγει μοναδικό αρχηγό και θα τερματίζει όταν όλες οι διεργασίες γνωρίζουν το UID του αρχηγού θα βασίζεται στον αλγόριθμο AsynchSpanningTree. Το δοσμένο σύστημα είναι ασύγχρονο, μη-κατευθυνόμενο συνεκτικό γράφημα και όλες οι διεργασίες έχουν UID, με μόνη γνώση τη διάμετρο.

Η κάθε διεργασία θα χαρακτηρίζεται από τα εξής:

1. Τη μεταβλητή parent, η οποία αρχικά θα είναι ίση με null και θα παίρνει την τιμή του πατέρα της διεργασίας στο επικαλυπτόμενο δέντρο που δημιουργείται.
2. Τη μεταβλητή leader, η οποία θα είναι αρχικά ίση με -1 για κάθε διεργασία και θα παίρνει την τιμή της ρίζας, δηλαδή του αρχηγού.
3. Έναν $buffer_{π}$ που θα περιέχει τα UID των παιδιών της κάθε διεργασίας και αρχικά θα είναι κενός.
4. Έναν $buffer_{n}$, ο οποίος θα περιέχει αρχικά όλους τους γείτονες της κάθε διεργασίας, και θα περιέχει όλους τους

γείτονες που δεν είναι ακόμη στο επικαλυπτόμενο δέντρο.

Ο αλγόριθμος θα λειτουργεί ως εξής:

Κάθε διεργασία θα εκτελεί μία παραλλαγή του AsynchSpanningTree και θα επιχειρεί να κατασκευάσει ένα επικαλυπτόμενο δέντρο με εκείνη ως ρίζα, δηλαδή ως αρχηγό. Αν δύο επικαλυπτόμενα δέντρα συναντηθούν, δηλαδή αν δύο κόμβοι που ανήκουν σε διαφορετικά δέντρα ανακαλύψουν ο ένας τον άλλον (χάρη στον $buffer_n$) τότε ο κόμβος του οποίου η ρίζα έχει το μικρότερο UID θα γίνει μέρος του άλλου επικαλυπτόμενου δέντρου, του οποίου η ρίζα έχει μεγαλύτερο UID. Με αυτόν τον τρόπο τελικά θα δημιουργηθεί ένα μοναδικό επικαλυπτόμενο δέντρο, του οποίου όλοι οι κόμβοι θα γνωρίζουν τη ρίζα, δηλαδή τον αρχηγό, που θα είναι αποθηκευμένη στη μεταβλητή leader.

Κάθε διεργασία θα έχει μία εσωτερική ενέργεια search(), τρεις διαφορετικές ενέργειες εισόδου ($receive_{i,j}$) και εξόδου($send_{i,j}$).

Αρχικά, κάθε διεργασία που θα επιχειρήσει να κατασκευάσει ένα δέντρο με εκείνη ρίζα (και συνεπώς αρχηγό), δεν θα έχει λάβει κάποιο μήνυμα και δεν θα έχει κάποιο γονέα και απλά θα θέσει τη μεταβλητή leader ίση με το UID της και την μεταβλητή parent ίση με τον εαυτό της. Έπειτα θα εκτελέσει τη search() που περιγράφεται παρακάτω.

Η search() θα εκτελείται ως ακολούθως:

Έστω δύο διεργασίες I,j. Αν ο $buffer_n$ της i δεν είναι κενός, τότε θα αφαιρέσει έστω τη διεργασία-γείτονα j από τον $buffer_n$ και θα της στείλει τη τιμή της μεταβλητής της leader και θα της πει ότι είναι το UID του δικού της αρχηγού (περίπτωση 1). Αν ο $buffer_n$ της i είναι κενός τότε αν δεν είναι αυτή η ρίζα του δέντρου (της οποίας η μεταβλητή parent είναι ίση με το UID

της), θα στείλει στον γονέα της στο επικαλυπτόμενο δέντρο, δηλαδή στον κόμβο που έχει αποθηκεύσει στη μεταβλητή parent, ότι είναι ο γονέας της και το UID της μεταβλητής leader της (περίπτωση 2). Αν ο $buffer_n$ της i είναι κενός και αυτή είναι η ρίζα του δέντρου (της οποίας η μεταβλητή parent είναι ίση με το UID της), τότε τερματίζει ο αλγόριθμος με αυτή τη διεργασία ως ρίζα του δέντρου και αρχηγό.

Ανάλογα με το μήνυμα που στέλνει η διεργασία i στη j, θα γίνονται και διαφορετικές ενέργειες από τη receive.

1. Αν η j λάβει από την i ότι της στέλνει τα στοιχεία του αρχηγού της (περίπτωση 1), τότε η j συγκρίνει την τιμή που δέχεται με την τιμή της δικιάς της μεταβλητής leader. Αν η τιμή της leader της είναι μικρότερη, τότε την αντικαθιστά με την εισερχόμενη, θέτει ως πατέρας της τη διεργασία i (με τη μεταβλητή parent), αδειάζει τον $buffer_\pi$ της, τοποθετεί στον $buffer_n$ της όλους τους γείτονες της εκτός από την i και εκτελεί και αυτή τη search(). Πρακτικά έτσι η j αλλάζει επικαλυπτόμενο δέντρο και πάει σε αυτό της i. Αν η τιμή της leader της j είναι ίση με την εισερχόμενη, τότε στέλνει στην i ότι είναι ήδη στο ίδιο δέντρο και της στέλνει την τιμή της μεταβλητής leader της (περίπτωση 3). Αν η τιμή της leader της j είναι μεγαλύτερη από την εισερχόμενη, τότε δεν κάνει τίποτα.
2. Αν η j λάβει από την i ότι είναι ο γονέας της (περίπτωση 2), τότε αν το UID που της έστειλε η i είναι ίσο με την τιμή της μεταβλητής leader της, προσθέτει την i στον $buffer_\pi$ της και εκτελεί τη search(). Αλλιώς αγνοεί το μήνυμα.
3. Αν η i δεχτεί από την j ότι είναι στο ίδιο δέντρο (περίπτωση 3) τότε αν το εισερχόμενο UID είναι ίσο με την τιμή της μεταβλητής leader της κάνει search(), αλλιώς αγνοεί το μήνυμα.

Ο παραπάνω αλγόριθμος που περιγράφεται είναι ορθός καθώς βασίζεται σε πολλές ταυτόχρονες εκτελέσεις μίας παραλλαγής του AsynchSpanningTree, ο οποίος είναι και αυτός ορθός, στην αρχή από πολλούς και τελικά από έναν μοναδικό κόμβο, ο οποίος θα είναι αυτός με το μεγαλύτερο UID. Τα μηνύματα και στις τρείς περιπτώσεις εξασφαλίζουν ότι μόνο το δέντρο με τη ρίζα με το μεγαλύτερο UID θα επεκτείνεται συνεχώς μέχρι να τερματιστεί ο αλγόριθμος.

Η πολυπλοκότητα επικοινωνίας του προτεινόμενου αλγορίθμου είναι O(nm), όπου n=|V| ο αριθμός των κόμβων και m=|E| ο αριθμός των ακμών. Αυτό συμβαίνει καθώς κάθε κόμβος θα στέλνει τουλάχιστον ένα μήνυμα σε κάθε γειτονικό του κόμβο και επειδή υπάρχουν συνολικά n κόμβοι, θα στέλνονται συνολικά mn μηνύματα.

Η χρονική πολυπλοκότητα είναι O(diam(l+d)) στη χειρότερη περίπτωση, όπου diam η διάμετρος του δικτύου, l άνω φράγμα στο χρόνο εκτέλεσης της search() και d άνω φράγμα στο χρόνο παράδοσης του μηνύματος, αφού ο χρόνος που χρειάζεται συνολικά για να τοποθετηθεί κάθε κόμβος στο επικαλυπτόμενο δέντρο με τη μεγαλύτερη ρίζα και να γνωρίζει το UID της είναι ίσος με τη διάμετρο του δικτύου, δηλαδή με τη μέγιστη απόσταση που υπάρχει στο δίκτυο.

Ερώτημα 2:

Σύμφωνα με τον αλγόριθμο RandomAttack, κάθε διεργασία σε κάθε μήνυμα που στέλνει επισυνάπτει και το key της, και την αρχική της τιμή και το επίπεδο πληροφόρησης της. Η απόφαση 1 λαμβάνεται μόνο όταν το επίπεδο της διεργασίας είναι ίσο με το key της και όλες οι αρχικές τιμές που έχει λάβει είναι 1.

Αν, όμως, υπάρχει κάποια διεργασία j με αρχική τιμή 0, τότε οποιαδήποτε διεργασία έχει παραλάβει μήνυμα από αυτή δεν πρόκειται να αποφασίσει 1, γιατί θα έχει λάβει ότι υπάρχει αρχική τιμή 0, και άρα η μόνη απόφαση είναι το 0.

Αν υποθέσουμε ότι δεν παραδοθεί σε καμία το μήνυμα της j, τότε, αφού κάθε διεργασία περιμένει να ακούσει από όλες πριν αλλάξει επίπεδο, καμία δεν θα πάει στο επίπεδο 1, πέρα από την j και αφού το key της κάθε μιας ανήκει στο διάστημα (1,r), θα είναι μεγαλύτερο του max $level_i$ της κάθε διεργασίας i και έτσι πάλι όλες θα αποφασίζουν 0.

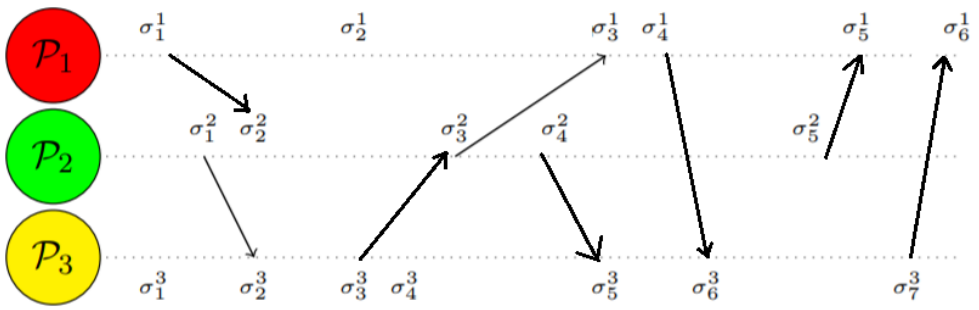
Άρα, σε κάθε περίπτωση, η ύπαρξη μίας διεργασίας με αρχική τιμή 0 θα έχει ως μόνη απόφαση το 0.

Η πιθανότητα συντονισμένης επίθεσης είναι ίση με P=1-ε, όπου ε η πιθανότητα διαφωνίας, αφού $P_{ολικό}$=P(διαφωνίας)+P(συμφωνίας)=1. Αν χάνεται το πολύ ένα μήνυμα τότε μόνο η διεργασία η οποία δεν το παρέλαβε θα μείνει πίσω ένα επίπεδο από τις υπόλοιπες, έστω j η διεργασία αυτή. Έτσι, όταν ληφθεί απόφαση στον γύρο r, το max $level_i$ για κάθε διεργασία i, εκτός της j, θα είναι ίσο με r (για την j θα είναι ίσο με r-1).

Οπότε, η μόνη περίπτωση που μπορεί να υπάρξει διαφωνία είναι αν key=max $level_i$, δηλαδή αν key=r. Επειδή το key παίρνει τιμές στο (1,r), η πιθανότητα να συμβεί αυτό είναι ίση με 1/r και άρα η πιθανότητα διαφωνίας ισούται με ε=1/r. Συνεπώς, η πιθανότητα συντονισμένης επίθεσης είναι ίση με P=1-ε=1-$\frac{1}{r}$=$\frac{r-1}{r}$.

Ερώτημα 3:

α. Μετά την προσθήκη δύο μηνυμάτων από κάθε διεργασία προς κάποια άλλη, το διάγραμμα μηνυμάτων είναι το ακόλουθο, όπου οι προσθήκες είναι οι πιο έντονες μαύρες γραμμές:



β. Η τοπική ιστορία της διεργασίας P_1 είναι η $h_1=\sigma_1^1,\sigma_2^1,\sigma_3^1,\sigma_4^1,\sigma_5^1,\sigma_6^1$.

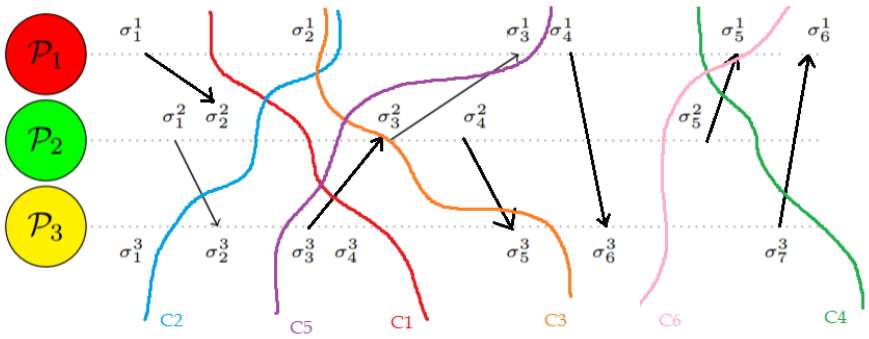
Η τοπική ιστορία της διεργασίας P_2 είναι η $h_2=\sigma_1^2,\sigma_2^2,\sigma_3^2,\sigma_4^2,\sigma_5^2$.

Η τοπική ιστορία της διεργασίας P_3 είναι η $h_3=\sigma_1^3,\sigma_2^3,\sigma_3^3,\sigma_4^3,\sigma_5^3,\sigma_6^3,\sigma_7^3$.

Η καθολική ιστορία του συστήματος Η είναι ίση με την ένωση των τοπικών ιστοριών των διεργασιών, δηλαδή $H=h_1 \cup h_2 \cup h_3$.

Άρα, η καθολική ιστορία είναι ίση με $H=\sigma_1^1,\sigma_2^1,\sigma_3^1,\sigma_4^1,\sigma_5^1,\sigma_6^1,$ $\sigma_1^2,\sigma_2^2,\sigma_3^2,\sigma_4^2,\sigma_5^2,$ $\sigma_1^3,\sigma_2^3,\sigma_3^3,\sigma_4^3,\sigma_5^3,\sigma_6^3,\sigma_7^3$.

γ. Το διάγραμμα μηνυμάτων μαζί με τις τομές είναι το ακόλουθο:



Οι τομές και τα σύνορα τους είναι όπως παρακάτω:

- $C_1=h_1^{\sigma_1^1} \cup h_2^{\sigma_2^2} \cup h_3^{\sigma_4^3} = \{\sigma_1^1, \sigma_1^2, \sigma_2^2, \sigma_1^3, \sigma_2^3, \sigma_3^3, \sigma_4^3\}$.
  Σύνορο τομής της C_1 είναι το $\{\sigma_1^1, \sigma_2^2, \sigma_4^3\}$.
- $C_2=h_1^{\sigma_2^1} \cup h_2^{\sigma_2^2} \cup h_3^{\sigma_1^3} = \{\sigma_1^1, \sigma_2^1, \sigma_1^2, \sigma_2^2, \sigma_1^3\}$.
  Σύνορο τομής της C_2 είναι το $\{\sigma_2^1, \sigma_2^2, \sigma_1^3\}$.
- $C_3=h_1^{\sigma_2^1} \cup h_2^{\sigma_2^2} \cup h_3^{\sigma_5^3} = \{\sigma_1^1, \sigma_2^1, \sigma_1^2, \sigma_2^2, \sigma_1^3, \sigma_2^3, \sigma_3^3, \sigma_4^3, \sigma_5^3\}$.
  Σύνορο τομής της C_3 είναι το $\{\sigma_2^1, \sigma_2^2, \sigma_5^3\}$.
- $C_4=h_1^{\sigma_4^1} \cup h_2^{\sigma_5^2} \cup h_3^{\sigma_7^3} = \{\sigma_1^1, \sigma_2^1, \sigma_3^1, \sigma_4^1, \sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2, \sigma_5^2, \sigma_1^3, \sigma_2^3, \sigma_3^3, \sigma_4^3, \sigma_5^3, \sigma_6^3, \sigma_7^3\}$.
  Σύνορο τομής της C_4 είναι το $\{\sigma_4^1, \sigma_5^2, \sigma_7^3\}$.
- $C_5=h_1^{\sigma_3^1} \cup h_2^{\sigma_2^2} \cup h_3^{\sigma_2^3} = \{\sigma_1^1, \sigma_2^1, \sigma_3^1, \sigma_1^2, \sigma_2^2, \sigma_1^3, \sigma_2^3\}$.
  Σύνορο τομής της C_5 είναι το $\{\sigma_3^1, \sigma_2^2, \sigma_2^3\}$.
- $C_6=h_1^{\sigma_5^1} \cup h_2^{\sigma_4^2} \cup h_3^{\sigma_6^3} = \{\sigma_1^1, \sigma_2^1, \sigma_3^1, \sigma_4^1, \sigma_5^1, \sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2, \sigma_1^3, \sigma_2^3, \sigma_3^3, \sigma_4^3, \sigma_5^3, \sigma_6^3\}$.
  Σύνορο τομής της C_6 είναι το $\{\sigma_5^1, \sigma_4^2, \sigma_6^3\}$.

Οι συνεπείς τομές είναι οι C_1,C_2,C_4 και οι μη συνεπείς οι C_3,C_5,C_6, όπως μπορούμε να προσδιορίσουμε και από τον τρόπο που τα βέλη “κόβουν” τις τομές. Αν η τομή είναι συνεπής, τα βέλη που θα την “κόβουν” θα ξεκινούν από τα αριστερά προς τα δεξιά.

Η C_1 είναι συνεπής: το βέλος $\sigma_3^3 \rightarrow \sigma_3^2$ κόβει την τομή από τα αριστερά προς τα δεξιά.

Η C_2 είναι συνεπής: το βέλος $\sigma_1^2 \rightarrow \sigma_2^3$ κόβει την τομή από τα αριστερά προς τα δεξιά.

Η C_4 είναι συνεπής: τα βέλη $\sigma_5^2 \rightarrow \sigma_5^1$ και $\sigma_7^3 \rightarrow \sigma_6^1$ κόβουν την τομή από τα αριστερά προς τα δεξιά.

Η C_3 δεν είναι συνεπής: το βέλος $\sigma_4^2 \rightarrow \sigma_5^3$ κόβει την τομή από τα δεξιά προς τα αριστερά.

Η C_5 δεν είναι συνεπής: το βέλος $\sigma_3^2 \rightarrow \sigma_3^1$ κόβει την τομή από τα δεξιά προς τα αριστερά.

Η C_6 δεν είναι συνεπής: το βέλος $\sigma_5^2 \rightarrow \sigma_5^1$ κόβει την τομή από τα δεξιά προς τα αριστερά.